ECE2020

IMAGE PROCESSING

(Human detection).

SSA LAB PROJECT

ECE2020

Table Of Content

- Introduction
- Method & Code
- Demo
- Conclusion

ECE2020

INTRODUCTION

*About the project:

-Our group's main focus in this project is using human detection.

-Instead of objects, we choose to detect 3 specific humans to detect for our image processing project.

-We choose 3 science fiction heroes from Marvel to detect: Thor, Iron man, Captain America.

METHOD& CODE

*Code:

-We use Python to build the main file for human detetion project.

Reasons:

+Data science

+Flexibility

+Helpful for System Automation & administration

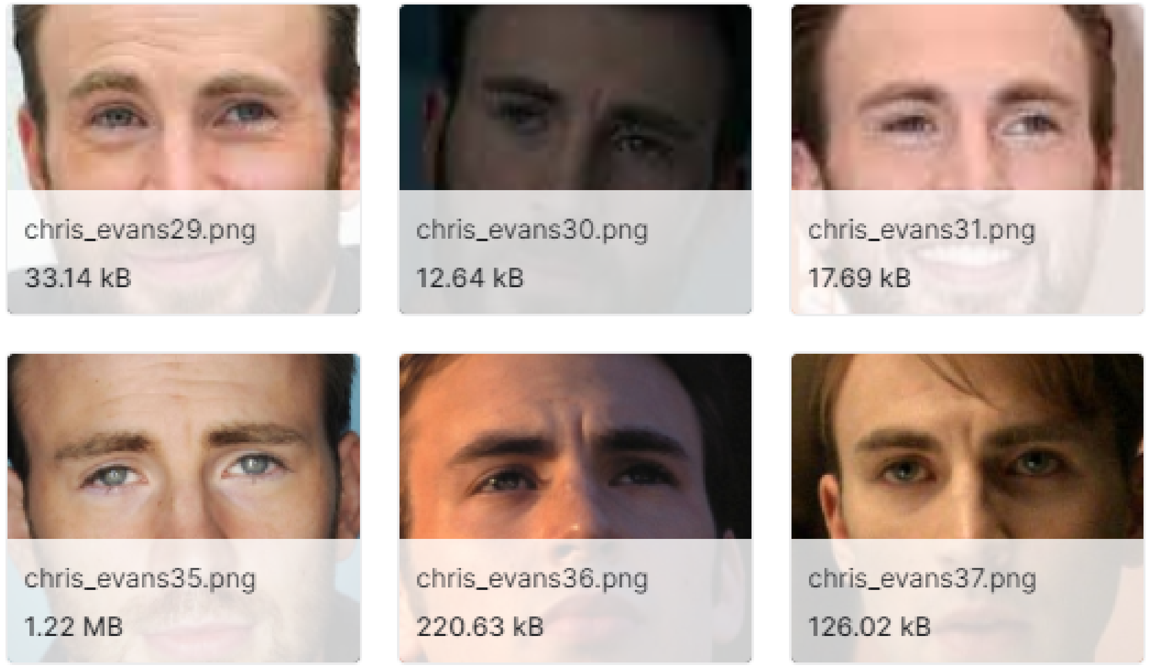
ECE2020

METHOD& CODE

3 types of files

- AUG file (Augmenting)
- Train file
- Detection file

dataset



chris_evans29.png
33.14 kB

chris_evans30.png
12.64 kB

chris_evans31.png
17.69 kB

chris_evans35.png
1.22 MB

chris_evans36.png
220.63 kB

chris_evans37.png
126.02 kB

METHOD& CODE

*AUG file:

-The AUG file main purpose is to process in order to increase the data. Different programs may use the AUG file type for different type of data.

METHOD& CODE

Code of AUG file:

```
import numpy as np
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.image_utils import array_to_img, img_to_array, load_img
import cv2
import glob

datagen = ImageDataGenerator(rotation_range =40,
                         width_shift_range = 0.2,
                         height_shift_range = 0.2,
                         shear_range=0.2,
                         zoom_range=0.2,
                         horizontal_flip = True,
                         fill_mode = 'nearest')

file = r"C:\Users\HP\Documents\Aug\*.jpg"
glob.glob(file)
images = [cv2.imread(image) for image in glob.glob(file)]
print(images)
for img in images:
    x = img_to_array(img)
    x = x.reshape((1,) + x.shape)

    i = 0
    for batch in datagen.flow (x, batch_size=1, save_to_dir =r'C:\Users\HP\Documents\Aug', save_prefix ='people2',
save_format='jpg'):
        i+=1
        if i > 10:
            break
```

METHOD& CODE

*Train file:

-This file is used to train the machine learning models. In this dataset, you will have the features (independent variables) and target (dependent variable) . Considering the loan prediction dataset, you will have features such as Gender, Age, Income, etc and the target is to predict loan status.

ECE2020

METHOD& CODE

Code of train file:

lib import

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers.legacy import Adam
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization, Conv2D, MaxPooling2D, Activation, Flatten, Dropout, Dense
from tensorflow.keras import backend as K
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import random
import cv2
import os
import glob
```

init training setting

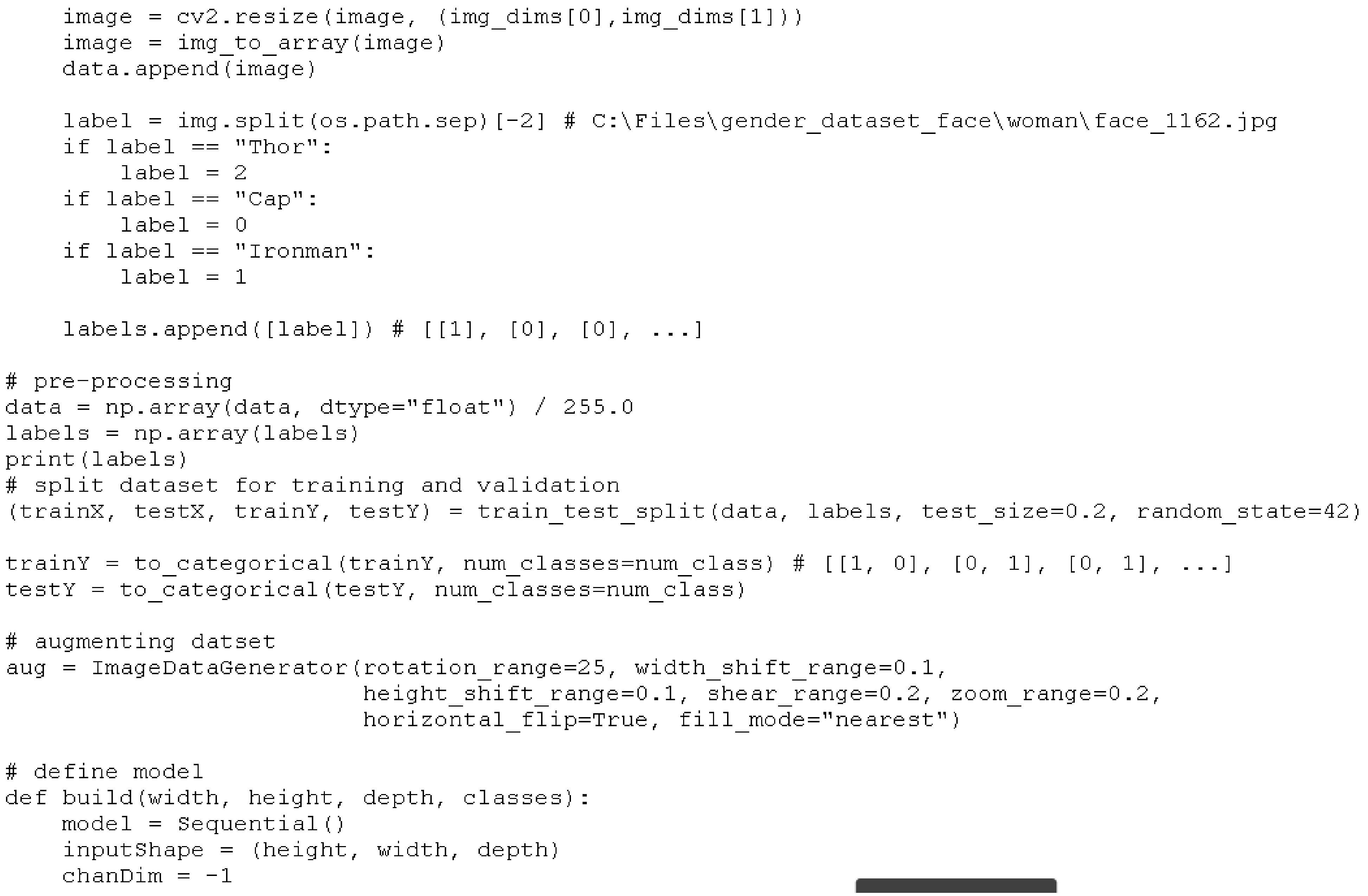
```
# initial parameters
epochs = 100
lr = 1e-3
batch_size = 16
img_dims = (96,96,3)
num_class = 3

data = []
labels = []
```

prepare training data

```
# load image files from the dataset
image_files = [f for f in glob.glob(r'C:\Users\penta\Documents\Crack\Dataset' + "/**/*", recursive=True) if not os.path.isdir(f)]
random.shuffle(image_files)

# converting images to arrays and labelling the categories
for img in image_files:

    image = cv2.imread(img)
```

ECE2020

METHOD& CODE



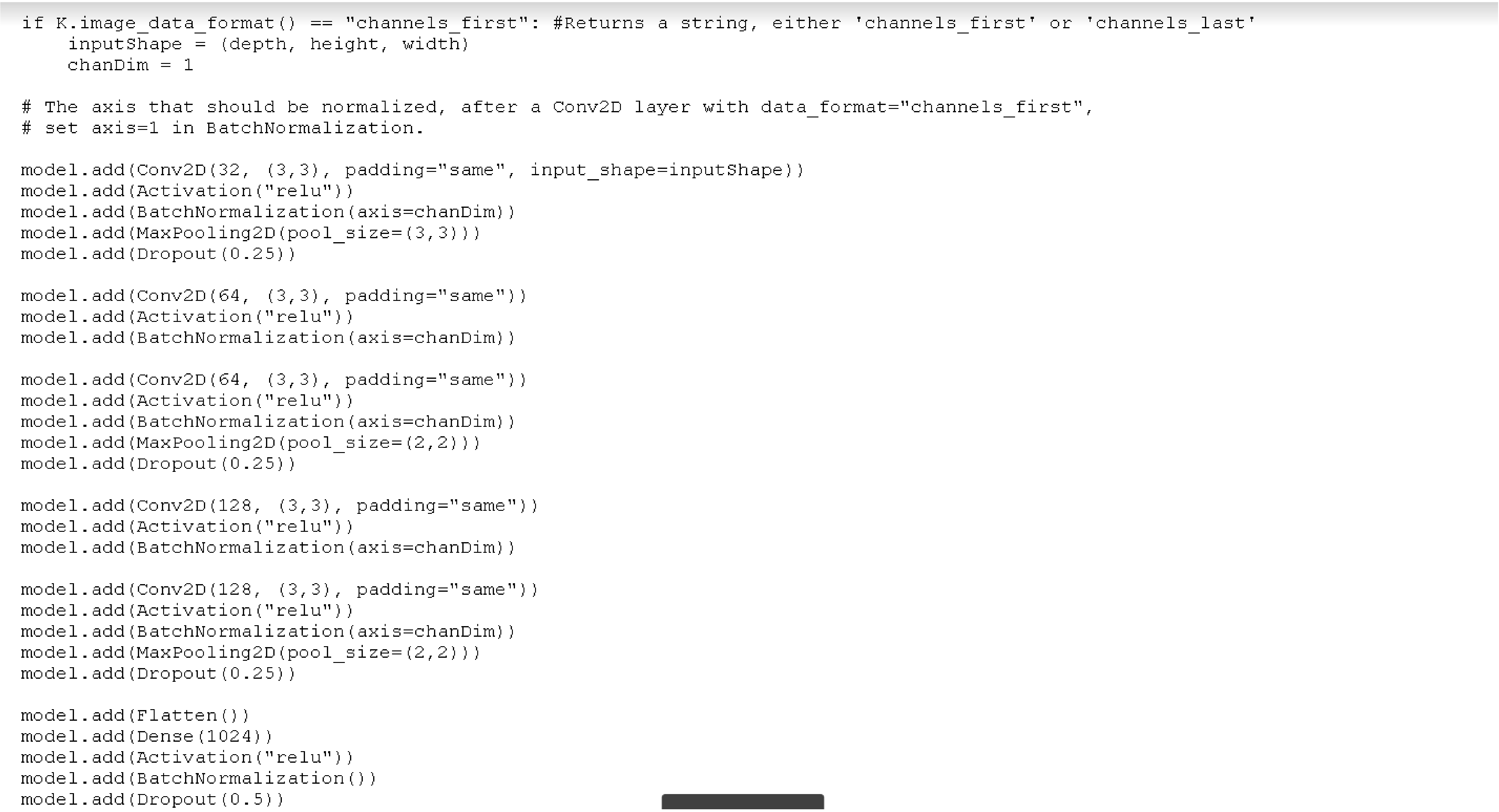
prepare training data

```
    image = cv2.resize(image, (img_dims[0],img_dims[1]))
    image = img_to_array(image)
    data.append(image)

    label = img.split(os.path.sep)[-2] # C:\Files\gender_dataset_face\woman\face_1162.jpg
    if label == "Thor":
        label = 2
    if label == "Cap":
        label = 0
    if label == "Ironman":
        label = 1

    labels.append([label]) # [[1], [0], [0], ...]
```

pre-process data

```
# pre-processing
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
print(labels)
# split dataset for training and validation
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2, random_state=42)

trainY = to_categorical(trainY, num_classes=num_class) # [[1, 0], [0, 1], [0, 1], ...]
testY = to_categorical(testY, num_classes=num_class)

# augmenting datset
aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
                         height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                         horizontal_flip=True, fill_mode="nearest")

# define model
def build(width, height, depth, classes):
    model = Sequential()
    inputShape = (height, width, depth)
    chanDim = -1
```

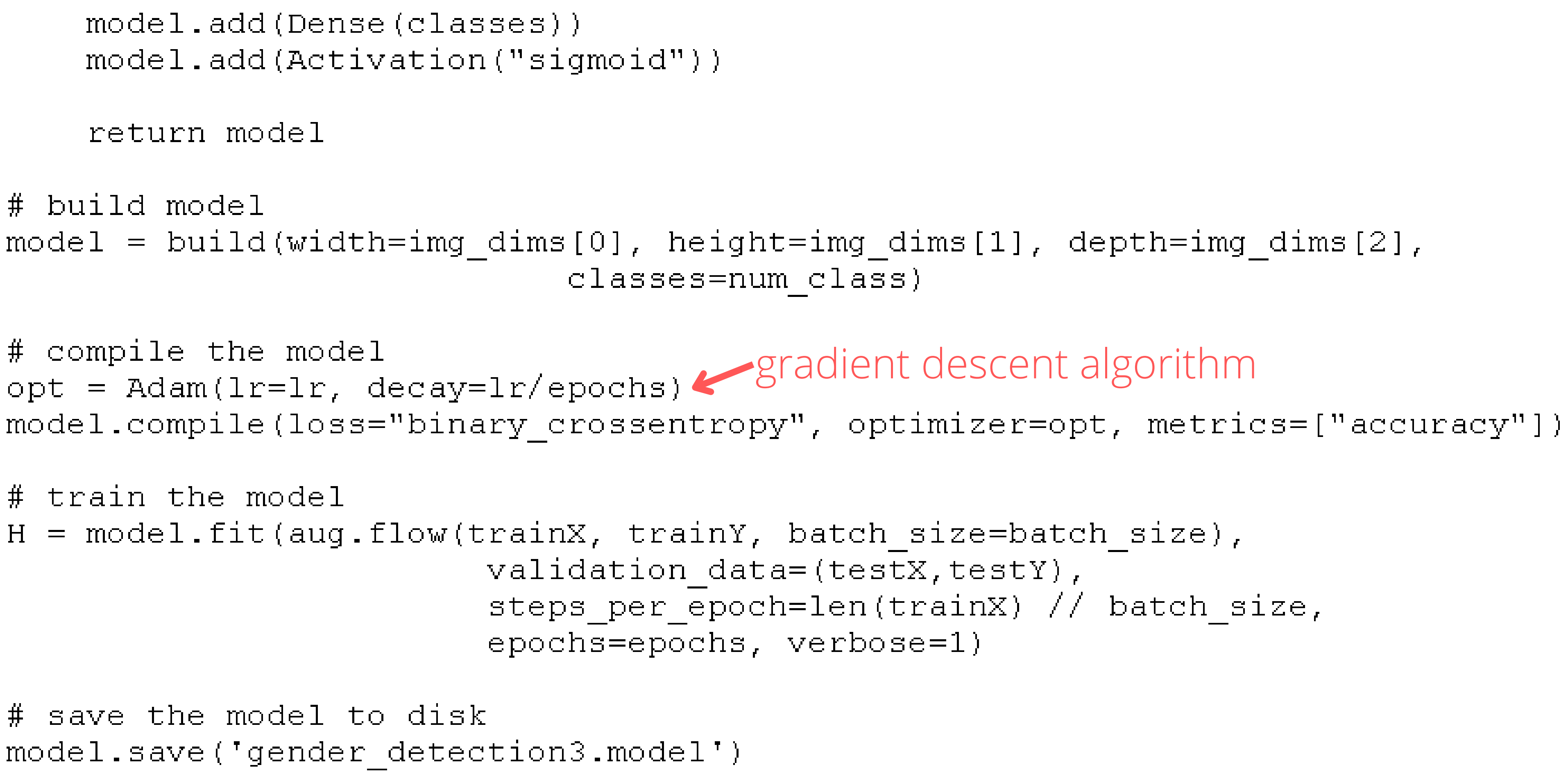
METHOD& CODE



```
if K.image_data_format() == "channels_first": #Returns a string, either 'channels_first' or 'channels_last'
    inputShape = (depth, height, width)
    chanDim = 1

# The axis that should be normalized, after a Conv2D layer with data_format="channels_first",
# set axis=1 in BatchNormalization.

model.add(Conv2D(32, (3,3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(64, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))

model.add(Conv2D(128, (3,3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

ECE2020

METHOD& CODE



```
    model.add(Dense(classes))
    model.add(Activation("sigmoid"))

    return model

# build model
model = build(width=img_dims[0], height=img_dims[1], depth=img_dims[2],
                            classes=num_class)

# compile the model
opt = Adam(lr=lr, decay=lr/epochs)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the model
H = model.fit(aug.flow(trainX, trainY, batch_size=batch_size),
                        validation_data=(testX,testY),
                        steps_per_epoch=len(trainX) // batch_size,
                        epochs=epochs, verbose=1)

# save the model to disk
model.save('gender_detection3.model')
```

gradient descent algorithm

train the model
and save to file

ECE2020

METHOD& CODE

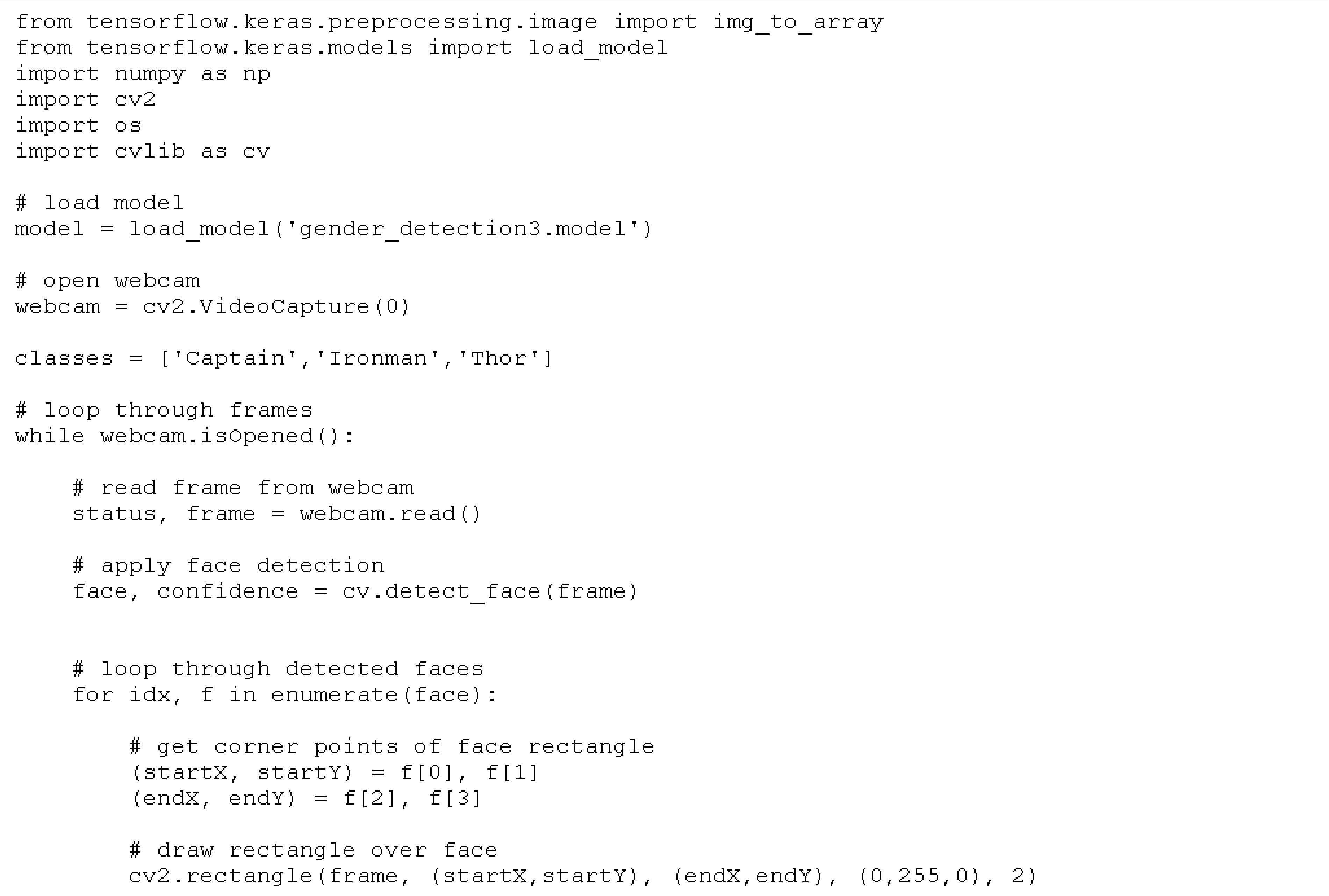
*Detection file:

-Finally, defection file is used to start and use the function of the program in our project.

ECE2020

METHOD& CODE

Code of detection file:

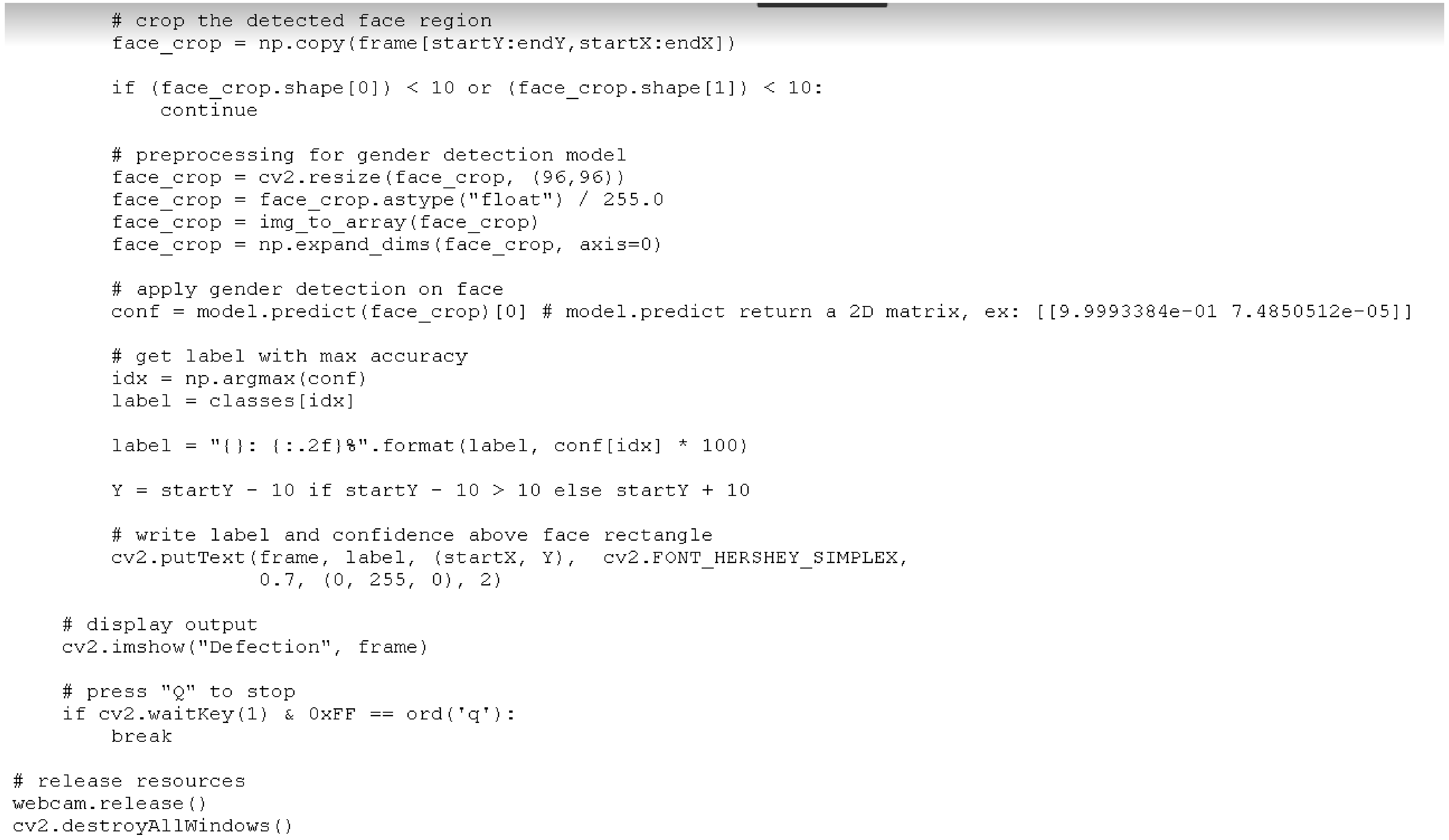


```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import cv2
import os
import cvlib as cv

# load model
model = load_model('gender_detection3.model')

# open webcam
webcam = cv2.VideoCapture(0)

classes = ['Captain','Ironman','Thor']

# loop through frames
while webcam.isOpened():

    # read frame from webcam
    status, frame = webcam.read()

    # apply face detection
    face, confidence = cv.detect_face(frame)


    # loop through detected faces
    for idx, f in enumerate(face):

        # get corner points of face rectangle
        (startX, startY) = f[0], f[1]
        (endX, endY) = f[2], f[3]

        # draw rectangle over face
        cv2.rectangle(frame, (startX,startY), (endX,endY), (0,255,0), 2)
```

ECE2020

METHOD& CODE

Code of detection file:



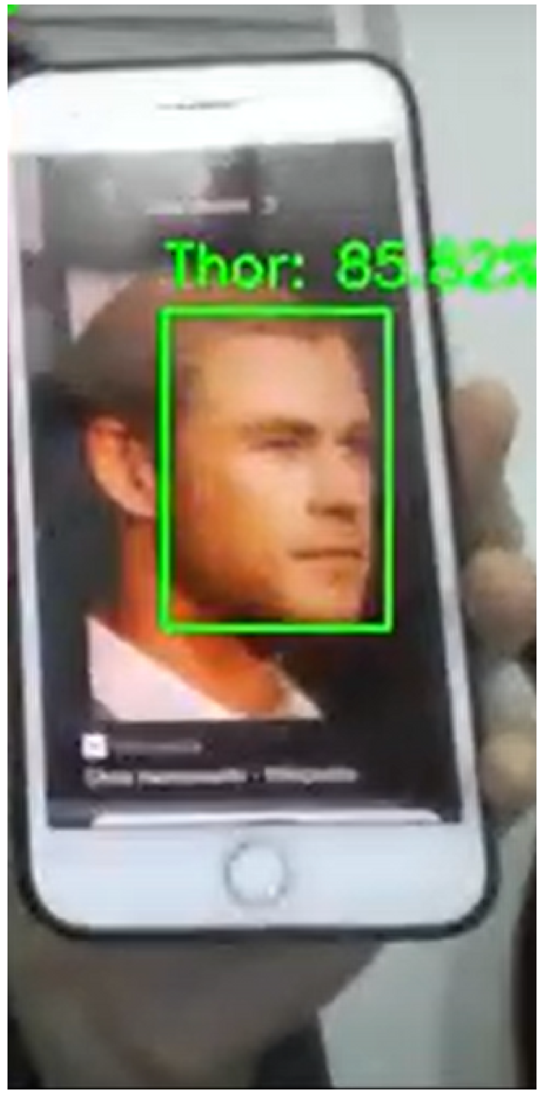
```
        # crop the detected face region
        face_crop = np.copy(frame[startY:endY,startX:endX])

        if (face_crop.shape[0]) < 10 or (face_crop.shape[1]) < 10:
            continue

        # preprocessing for gender detection model
        face_crop = cv2.resize(face_crop, (96,96))
        face_crop = face_crop.astype("float") / 255.0
        face_crop = img_to_array(face_crop)
        face_crop = np.expand_dims(face_crop, axis=0)

        # apply gender detection on face
        conf = model.predict(face_crop)[0] # model.predict return a 2D matrix, ex: [[9.9993384e-01 7.4850512e-05]]

        # get label with max accuracy
        idx = np.argmax(conf)
        label = classes[idx]

        label = "{}: {:.2f}%".format(label, conf[idx] * 100)

        Y = startY - 10 if startY - 10 > 10 else startY + 10

        # write label and confidence above face rectangle
        cv2.putText(frame, label, (startX, Y),  cv2.FONT_HERSHEY_SIMPLEX,
                    0.7, (0, 255, 0), 2)

    # display output
    cv2.imshow("Defection", frame)

    # press "Q" to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release resources
webcam.release()
cv2.destroyAllWindows()
```

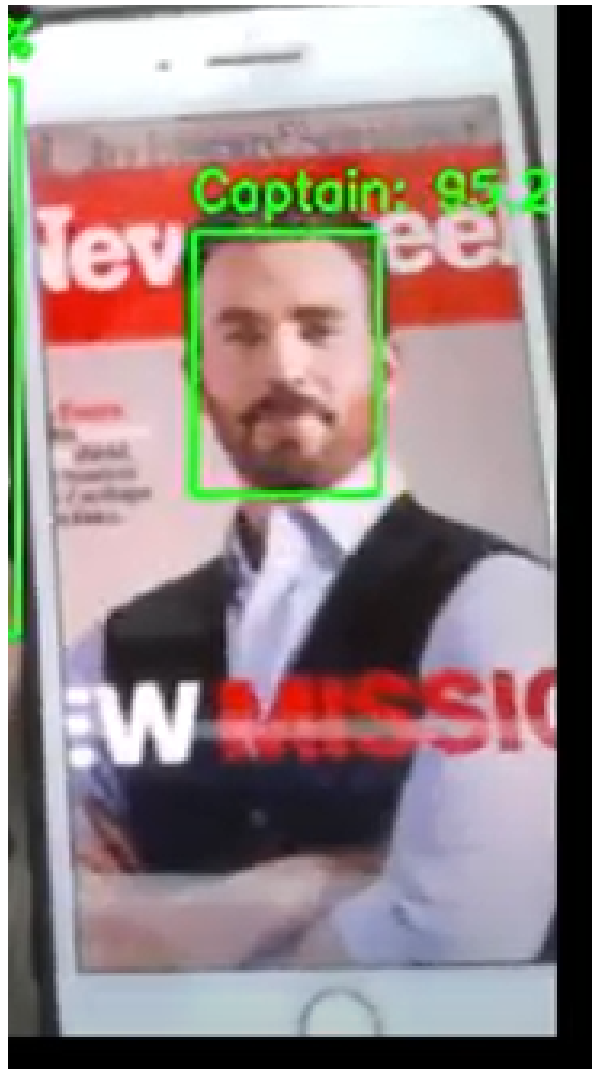
ECE2020

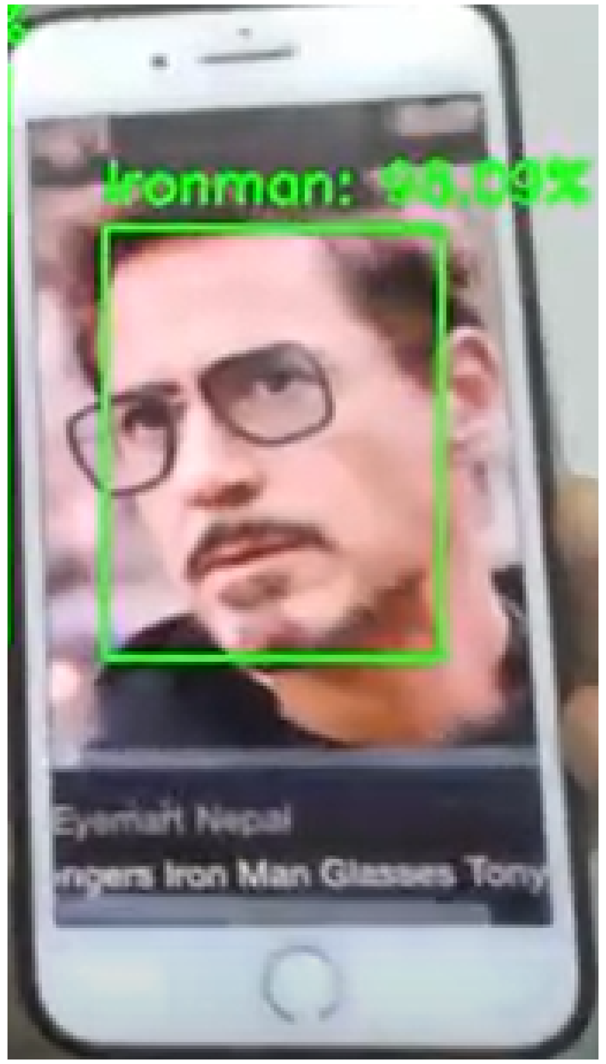
DEMO

-Here is the result when we run the project's program

-It can easily detect 3 different targets that we want to detect in our project.







ECE2020

CONCLUSION

Pros:

+Its can work quite accurate.

+Fast approximation ability and detection rates.

Cons:

+The program still quite simple .

+Not 100% accuracy

==> Can be developed more in code and method to improve the program efficiency.

Thank you!